Rigel - DriverPlan

PLANO DE TESTES

Versão 1.0

Tabela - Integrantes do Grupo:

Mat.	Nome	Função (responsabilidade)
221008786	*Mateus Villela Consorte*	*Product Owner*
211062179	*Marcelo de Araújo Lopes*	*Scrum Master*
211062473	*Samuel Afonso da Silva Santos*	*Dev Back End*
221008679	*Pablo Serra Carvalho*	*Dev Back End*
221031120	*Arthur Fonseca Vale*	*Dev Back End*
202045820	*Karolina Vieira Barbosa*	*Dev Front End*
221037803	*Letícia Kellen Ramos Paiva*	*Dev Front End*
202017067	*Raul Falluh F. de Mendonça*	*Dev Front End*

Histórico de Revisões

Data	Versão	Descrição	Autor
01/06/2024	*1.0*		

PLANO DE TESTES

1. PROPÓSITO

O objetivo deste plano de testes é definir a estratégia abrangente, a metodologia, o cronograma e os recursos necessários para validar a qualidade e o desempenho do software de gerenciamento de viagens para motoristas privados DriverPlan. Este plano visa garantir que todas as funcionalidades do sistema estejam de acordo com os requisitos especificados e que o sistema funcione corretamente em diferentes condições. Sendo assim:

1.1. Validar a funcionalidade completa do DriverPlan conforme especificado nos requisitos.

1.2. Identificar e corrigir defeitos antes da implantação do sistema.

1.3. Garantir que o software seja robusto, confiável e seguro para uso dos motoristas e clientes.

2. ESCOPO

O escopo deste plano de testes abrange todas as atividades necessárias para garantir que o Sistema XYZ funcione conforme os requisitos especificados e atenda às expectativas dos usuários. Este plano cobre os seguintes componentes e áreas:

3. REFERÊNCIAS

Para garantir que os testes sejam conduzidos de maneira adequada e alinhada com os requisitos e especificações do DriverPlan, os seguintes documentos são referenciados (links):

- **Documento de Visão do produto e do projeto;**
- **Documento de Arquitetura;**
- **Sumário Executivo;**
- **Casos de uso do projeto.**

4. REQUISITOS A TESTAR

4.1. **Testes de Login e Logout:** Verificar se os usuários podem acessar o sistema corretamente.

4.2. **Testes de Permissões:** Garantir que as permissões de acesso sejam aplicadas corretamente.

4.3. **Cadastro de Motoristas e Clientes:** Verificar se os usuários podem ser cadastrados corretamente.

4.4. **Criação de Cronogramas**: Verificar se os motoristas podem criar e modificar cronogramas de viagens.

4.5. **Validação de Conflitos**: Testar a prevenção de conflitos de horários entre viagens.

4.6. **Cálculo de Rotas**: Verificar se as rotas são calculadas corretamente com base nos dados do Google Maps.

4.7. **Precisão das Informações**: Testar a precisão das informações de distância e tempo fornecidas pelo Google Maps.

4.8. **Cálculo Automático de Tarifas**: Verificar se as tarifas são calculadas corretamente com base nos parâmetros definidos.

5. ESTRATÉGIAS E FERRAMENTAS DE TESTES

5.1. Tipos de testes

5.1.1. **Testes Unitários:** Testam unidades isoladas do código, como funções ou métodos individuais.

5.1.2. **Testes de Integração:** Testam a integração de diferentes partes do sistema, verificando se elas funcionam juntas corretamente.

5.1.3. **Testes de Modelo:** Testam os modelos do Django, incluindo métodos e propriedades personalizadas.

5.1.4. **Testes de Formulário:** Testam a lógica dos formulários, incluindo validações, permissões e métodos personalizados.

5.1.5. **Testes de Visualização:** Testam as visualizações para garantir que elas retornem às respostas corretas e renderizar os templates esperados.

5.1.6. **Testes de Template:** Verificam se os templates estão renderizando corretamente e se contêm o conteúdo esperado.

5.1.7. **Testes de URL:** Testam as configurações de URL para garantir que as rotas estão mapeadas corretamente para as visualizações apropriadas.

5.1.8. **Testes de Cliente:** Utilizam o cliente de teste do Django para simular requisições HTTP e verificar as respostas.

5.1.9. **Testes Funcionais:** Testam a aplicação como um todo do ponto de vista do usuário, simulando interações completas.

5.2. Ferramentas de testes

5.2.1. Pytest:

- **Vantagens:** Possui simplicidade, escalabilidade e recursos poderosos, como suporte de fixtures e parametrização, sintaxe concisa e um rico ecossistema de plugins.

- **Desvantagens:** Embora o PyTest seja relativamente fácil de aprender, a compreensão de recursos e acessórios avançados pode levar algum tempo para os recém-chegados e é necessário uma instalação externa.

5.2.2. Unittest:

- **Vantagens:** Adotado generalizadamente e faz parte da Biblioteca Padrão Python. Possui descoberta de teste, suporte de fixture e isolamento de teste consistente.

- **Desvantagens:** Possui uma sintaxe mais detalhada e maior complexidade para grandes projetos demandando muito tempo de investimento além de suporte de parametrização limitado.

6. CRITÉRIOS DE ACEITAÇÃO

Os critérios de aceitação são condições específicas que uma funcionalidade deve atender para ser considerada completa e pronta para entrega. Eles são definidos durante o planejamento da sprint em colaboração entre o Product Owner, a equipe de desenvolvimento e outros stakeholders relevantes. A importância dos critérios de aceitação inclui:

6.1. Documentação

6.2. Autenticação de Usuário

6.3. Integração com APIs Externas

7. CONCLUSÃO

Em suma, o plano de testes do DriverPlan não apenas garante a qualidade técnica do sistema, mas também reforça nosso compromisso com a excelência e a satisfação do cliente. Estamos confiantes de que os esforços investidos em testes resultarão em um sistema robusto, confiável, seguro e pronto para superar as expectativas dos usuários, contribuindo positivamente para a gestão eficiente e segura de viagens.